

**This Page Is Inserted by IFW Operations
and is not a part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- **BLACK BORDERS**
- **TEXT CUT OFF AT TOP, BOTTOM OR SIDES**
- **FADED TEXT**
- **ILLEGIBLE TEXT**
- **SKEWED/SLANTED IMAGES**
- **COLORED PHOTOS**
- **BLACK OR VERY BLACK AND WHITE DARK PHOTOS**
- **GRAY SCALE DOCUMENTS**

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-161283
(43)Date of publication of application : 21.06.1996

(51)Int.Cl.

G06F 15/177

(21)Application number : 06-330694
(22)Date of filing : 07.12.1994

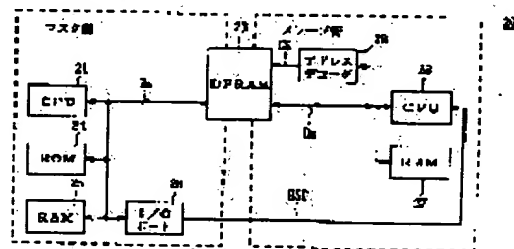
(71)Applicant : SONY CORP
(72)Inventor : KAMEYAMA NAOKI
SUGINO AKINOBU
YASUI HIROYUKI

(54) PLURAL PROCESSOR SYSTEMS

(57)Abstract:

PURPOSE: To embody plural processor systems whose constitutions can be further simplified.

CONSTITUTION: By making a slave processor 22 a reset state by a reset means 26, reading the boot code of the slave processor 22 from a memory 24, writing the code in the area corresponding to the boot code storage address of a shared memory 23, making the slave processor 22 read the boot code on the shared memory 23 by releasing the reset state of the slave processor 22 and starting the slave processor 22, the reading exclusive memory which has been required on a slave side to store the precode of a conventional slave processor 22 can be reduced. As a result, the constitution can be further simplified.



LEGAL STATUS

[Date of request for examination] 06.06.2001
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁(JP)

(12) 公開特許公報(A)

(11)特許出願公開番号

特開平8-161283

(43)公開日 平成8年(1996)6月21日

(51)Int.Cl.⁶

識別記号

庁内整理番号

FI

技術表示箇所

G 0 6 F 15/177

G 0 6 F 15/ 16

4 2 0 S

審査請求 未請求 請求項の数3 FD (全 9 頁)

(21)出願番号 特願平6-330694

(22)出願日 平成6年(1994)12月7日

(71)出願人 000002185

ソニー株式会社

東京都品川区北品川6丁目7番35号

(72)発明者 亀山 直樹

東京都品川区北品川6丁目7番35号ソニー株式会社内

(72)発明者 杉野 彰信

東京都品川区北品川6丁目7番35号ソニー株式会社内

(72)発明者 安居 宏之

東京都品川区北品川6丁目7番35号ソニー株式会社内

(74)代理人 弁理士 田辺 恵基

(54)【発明の名称】 複数プロセッサシステム

(57)【要約】

【目的】本発明は複数プロセッサシステムに関し、一段と構成を簡易にし得る複数プロセッサシステムを実現する。

【構成】リセット手段26によつてスレーブプロセッサ22をリセット状態にし、メモリ24からスレーブプロセッサ22のブートコードを読み出して共有メモリ23のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセッサ22のリセット状態を解除することによつてスレーブプロセッサ22に共有メモリ23上のブートコードを読み出させて当該スレーブプロセッサ22を立ち上げることにより、従来スレーブプロセッサ22のブートコードを格納しておくためスレーブ側に必要だった読み出し専用メモリを削減することができ、これにより一段と構成を簡易にできる。

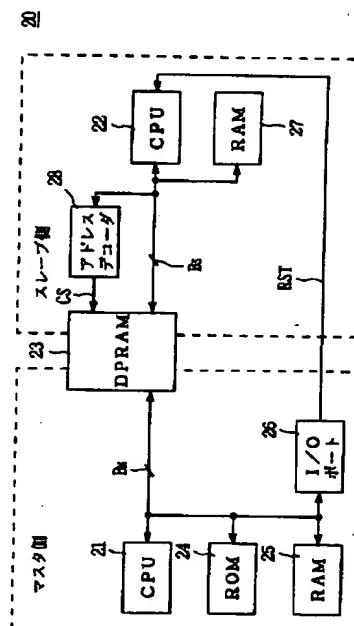


図1 複数CPUシステムの構成

1

【特許請求の範囲】

【請求項1】 マスタプロセッサと少なくとも1つのスレーブプロセッサとを有し、当該マスタプロセッサとスレーブプロセッサとを共有メモリを介して接続してなる複数プロセッサシステムにおいて、

上記マスタプロセッサの制御に応じて上記スレーブプロセッサをリセット状態にさせるリセット手段と、
上記共有メモリを、上記スレーブプロセッサのブートコード格納アドレスを含むアドレス領域に割り当てるアドレスデコーダと、

上記スレーブプロセッサのブートコードが格納され、上記マスタプロセッサによって読み出し可能なメモリとを
具え、

上記リセット手段によって上記スレーブプロセッサをリセット状態にし、上記マスタプロセッサによって上記メモリから上記スレーブプロセッサのブートコードを読み出して上記共有メモリの上記ブートコード格納アドレスに対応する領域に書き込み、上記スレーブプロセッサのリセット状態を解除することによって上記スレーブプロセッサに上記共有メモリ上のブートコードを読み出させて当該スレーブプロセッサを立ち上げることを特徴とする複数プロセッサシステム。

【請求項2】 マスタプロセッサと少なくとも1つのスレーブプロセッサとを有し、当該マスタプロセッサとスレーブプロセッサとを共有メモリを介して接続してなる複数プロセッサシステムにおいて、

上記マスタプロセッサの制御に応じて上記スレーブプロセッサをリセット状態にさせるリセット手段と、

上記共有メモリを、上記スレーブプロセッサのブートコード格納アドレスを含むアドレス領域に割り当てるアドレスデコーダと、

上記スレーブプロセッサによって書き込み及び読み出し可能な第1のメモリと、

上記マスタプロセッサが上記共有メモリに書き込んだプログラムコードを読み出して上記第1のメモリに格納する命令内容でなる上記スレーブプロセッサのブートコード及び上記スレーブプロセッサのプログラムコードが格納され、上記マスタプロセッサによって読み出し可能な第2のメモリと、

を具え、

上記リセット手段によって上記スレーブプロセッサをリセット状態にし、上記マスタプロセッサによって上記第2のメモリから上記スレーブプロセッサのブートコードを読み出して上記共有メモリの上記ブートコード格納アドレスに対応する領域に書き込み、上記スレーブプロセッサのリセット状態を解除することによって上記スレーブプロセッサに上記共有メモリ上のブートコードを読み出させて当該スレーブプロセッサを立ち上げ、さらに上記スレーブプロセッサに上記ブートコードの命令内容を実行させることによって、上記マスタプロセッサが上記

2

第2のメモリから読み出して上記共有メモリ上に書き込んだプログラムコードを当該共有メモリから読み出して上記第1のメモリに格納することを特徴とする複数プロセッサシステム。

【請求項3】 上記マスタプロセッサが上記第2のメモリから読み出して上記共有メモリ上に書き込んだプログラムコードを当該共有メモリから読み出して上記第1のメモリに格納する処理を繰り返し実行することを特徴とする請求項2に記載の複数プロセッサシステム。

10 【発明の詳細な説明】

【0001】

【目次】以下の順序で本発明を説明する。

産業上の利用分野

従来の技術（図4）

発明が解決しようとする課題

課題を解決するための手段（図1～図3）

作用（図1～図3）

実施例（図1～図3）

発明の効果

20 【0002】

【産業上の利用分野】 本発明は複数プロセッサシステムに関し、例えば複数のCPU（Central Processing Unit）を有するシステムに適用して好適なものである。

【0003】

【従来の技術】 従来、複数のCPUを有するシステム（以下これを複数CPUシステムと呼ぶ）においては、各CPU間のインターフェースとして入出力ポートを2つ持ついわゆるデュアルポートRAM（Random Access Memory）が一般的に用いられている。

30 【0004】 例えば図4に示すように、複数CPUシステム1では、マスタ側のCPU2とスレーブ側のCPU3、4との間にそれぞれデュアルポートRAM（DPRAM）5、6を設け、当該デュアルポートRAM5、6をマスタ側とスレーブ側の両方でアクセスすることによりマスタ側とスレーブ側との間でデータ等を受け渡すようになされている。ここで各CPU2～4に対してはそれぞれ不揮発性メモリのROM（Read Only Memory）7～9が設けられており、各CPU2～4はそれぞれこのROM7～9に格納されたプログラムコードに基づいて動作する。また各CPU2～4に対してはそれぞれ揮発性メモリのRAM10～12が設けられており、各CPU2～4はそれぞれこのRAM10～12に対して種々のデータを読み書きする。

40 【0005】

【発明が解決しようとする課題】 ところで複数CPUシステム1においては、スレーブ側のCPU3、4を動作させるプログラムコードの大部分をマスタ側のROM7に格納しておき、そのプログラムコードをそれぞれデュアルポートRAM5、6を介してRAM11、12にコピーすることにより、ROM8、9の容量を比較的小さ

くすることができると考えられる。

【0006】しかしながら複数CPUシステム1では、上述のようにスレーブ側のCPU3、4を動作させるプログラムコードの大部分をマスタ側からコピーするようにしたとしても、CPU3、4のブートコード（すなわちCPU3、4がリセット直後に読み出して立ち上がるために必要なコード）やCPU3、4がそれぞれデュアルポートRAM5、6を介してデータを送受信するときに必要なプログラムコードを格納するためにROM8、9がどうしても必要である。すなわちROM8、9の容量を小さくすることはできるが、比較的高価なROM8、9を完全になくすることができず、この分全体として構成を簡易にし得ないと共に、コスト的に高くなる問題がある。

【0007】またデュアルポートRAMを用いて各CPU間でデータ通信するものとしてこの他にも、特開平1-312659号公報に記載されるものがある。これに記載される複数CPUシステムでは、メインCPUとトランスCPUとの間でステータス情報を通信する場合、送信側はデュアルポートRAM上の所定のステータス情報を変更すると共に、変更したステータス情報を指示するヘッダをセットし、受信側は変更されたステータス情報をヘッダによつて分析して当該変更されたステータス情報を読み出すようになされている。これによりこの複数CPUシステムでは、1回の通信で同時に複数のステータス情報を変更して送信でき、処理時間を低減することができる。しかしながらこの複数CPUシステムでも、通信する際に必要なプログラムコードを格納するためにROMを必要とし、この分全体として構成を簡易にし得ないと共に、コスト的に高くなる問題がある。

【0008】本発明は以上の点を考慮してなされたもので、一段と構成を簡易にし得る複数プロセッサシステムを提案しようとするものである。

【0009】

【課題を解決するための手段】かかる課題を解決するため本発明においては、マスタプロセッサ21と少なくとも1つのスレーブプロセッサ22とを有し、当該マスタプロセッサ21とスレーブプロセッサ22とを共有メモリ23を介して接続してなる複数プロセッサシステム20において、マスタプロセッサ21の制御に応じてスレーブプロセッサ22をリセット状態にさせるリセット手段26と、共有メモリ23を、スレーブプロセッサ22のブートコード格納アドレスを含むアドレス領域に割り当てるアドレスデコーダ28と、スレーブプロセッサ22のブートコードが格納され、マスタプロセッサ21によつて読み出し可能なメモリ24とを設け、リセット手段26によつてスレーブプロセッサ22をリセット状態にし、マスタプロセッサ21によつてメモリ24からスレーブプロセッサ22のブートコードを読み出して共有メモリ23のブートコード格納アドレスに対応する領域

に書き込み、スレーブプロセッサ22のリセット状態を解除することによつてスレーブプロセッサ22に共有メモリ23上のブートコードを読み出させて当該スレーブプロセッサ22を立ち上げるようにした。

【0010】また本発明においては、マスタプロセッサ21と少なくとも1つのスレーブプロセッサ22とを有し、当該マスタプロセッサ21とスレーブプロセッサ22とを共有メモリ23を介して接続してなる複数プロセッサシステム20において、マスタプロセッサ21の制御に応じてスレーブプロセッサ22をリセット状態にさせるリセット手段26と、共有メモリ23を、スレーブプロセッサ22のブートコード格納アドレスを含むアドレス領域に割り当てるアドレスデコーダ28と、スレーブプロセッサ22によつて書き込み及び読み出し可能な第1のメモリ27と、マスタプロセッサ21が共有メモリ23に書き込んだプログラムコードを読み出して第1のメモリ27に格納する命令内容でなるスレーブプロセッサ22のブートコード及びスレーブプロセッサ22のプログラムコードが格納され、マスタプロセッサ21によつて読み出し可能な第2のメモリ24とを設け、リセット手段26によつてスレーブプロセッサ22をリセット状態にし、マスタプロセッサ21によつて第2のメモリ24からスレーブプロセッサ22のブートコードを読み出して共有メモリ23のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセッサ22のリセット状態を解除することによつてスレーブプロセッサ22に共有メモリ23上のブートコードを読み出させて当該スレーブプロセッサ22を立ち上げ、さらにスレーブプロセッサ22にブートコードの命令内容を実行させることによつて、マスタプロセッサ21が第2のメモリ24から読み出して共有メモリ23上に書き込んだプログラムコードを当該共有メモリ23から読み出して第1のメモリ27に格納するようにした。

【0011】また本発明においては、マスタプロセッサ21が第2のメモリ24から読み出して共有メモリ23上に書き込んだプログラムコードを当該共有メモリ23から読み出して第1のメモリ27に格納する処理を繰り返し実行するようにした。

【0012】

【作用】リセット手段26によつてスレーブプロセッサ22をリセット状態にし、マスタプロセッサ21によつてメモリ24からスレーブプロセッサ22のブートコードを読み出して共有メモリ23のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセッサ22のリセット状態を解除することによつてスレーブプロセッサ22に共有メモリ23上のブートコードを読み出させて当該スレーブプロセッサ22を立ち上げるようにしたことにより、従来スレーブプロセッサ22のブートコードを格納しておくためスレーブ側に必要だった読み出し専用メモリを削減することができる。

【0013】またリセット手段26によつてスレーブプロセッサ22をリセット状態にし、マスタプロセッサ21によつて第2のメモリ24からスレーブプロセッサ22のブートコードを読み出して共有メモリ23のブートコード格納アドレスに対応する領域に書き込み、スレーブプロセッサ22のリセット状態を解除することによつてスレーブプロセッサ22に共有メモリ23上のブートコードを読み出させて当該スレーブプロセッサ22を立ち上げ、さらにスレーブプロセッサ22にブートコードの命令内容を実行させることによつて、マスタプロセッサ21が第2のメモリ24から読み出して共用メモリ23上に書き込んだプログラムコードを当該共用メモリ23から読み出して第1のメモリ27に格納するようにしたことにより、第2のメモリ24に格納されたスレーブプロセッサ22のプログラムコードを第1のメモリ27にコピーできると共に、従来スレーブプロセッサ22のブートコードやプログラムコードを格納しておくためスレーブ側に必要だった読み出し専用メモリを削減することができる。

【0014】またマスタプロセッサ21が第2のメモリ24から読み出して共用メモリ23上に書き込んだプログラムコードを当該共用メモリ23から読み出して第1のメモリ27に格納する処理を繰り返し実行するようにしたことにより、共用メモリ23の容量よりも大きなプログラムコードを第1のメモリ27にコピーできる。

【0015】

【実施例】以下図面について、本発明の一実施例を詳述する。

【0016】図1において、20は全体として本発明を適用した複数CPUシステムを示し、マスタ側のCPU21とスレーブ側のCPU22との間にデュアルポートRAM(DPRAM)23を設け、当該デュアルポートRAM23をマスタ側とスレーブ側の両方でアクセスすることによりマスタ側とスレーブ側の間で種々のコードやデータを受け渡すようになされている。

【0017】この実施例の場合、マスタ側には不揮発性メモリのROM24が設けられ、このROM24にマスタ側のCPU21を動作させるプログラムコードやスレーブ側のCPU22を動作させるプログラムコード(リセット解除後にCPU22が読み出して実行するブートコードを含む)が格納されている。またマスタ側には揮発性メモリのRAM25が設けられており、マスタ側のCPU25はこのRAM25を作業領域として種々のデータを読み書きする。さらにマスタ側にはCPU21の指示に応じてリセット信号RSTを生成する入出力ポートいわゆるI/Oポート26が設けられており、このリセット信号RSTをスレーブ側のCPU22に対して出力して当該CPU22をリセット状態にするようになされている。この場合、CPU21、デュアルポートRAM23、ROM24、RAM25及びI/Oポート26

はそれぞれバスB_sによつて接続され、このバスB_sを介して書き込み制御信号、読み出し制御信号、セレクト信号、アドレス信号、データ等の種々の情報を受け渡すようになされている。

【0018】これに対してスレーブ側には揮発性メモリのRAM27が設けられており、デュアルポートRAM23を介してコピーしたCPU22を動作させるためのプログラムコードをこのRAM27に格納するようになされている。またRAM27はCPU22の作業領域としても使用され、種々のデータが読み書きされる。さらにスレーブ側には簡単なPLD(Programmable Logic Device)で構成されたアドレスデコーダ28が設けられており、このアドレスデコーダ28によつてCPU22のアドレス信号をデコードし、その結果得た信号をチップセレクト信号CSとしてデュアルポートRAM23に供給することにより、当該デュアルポートRAM23をブートコード格納アドレスを含む領域に割り当てるようになされている。この場合、CPU22、RAM27、デュアルポートRAM23及びアドレスデコーダ28はそれぞれバスB_sによつて接続され、このバスB_sを介して書き込み制御信号、読み出し制御信号、セレクト信号、アドレス信号、データ等の種々の情報を受け渡すようになされている。

【0019】ここでデュアルポートRAM23を例えば16[Kbit]のもの2つで構成し、RAM27を例えば256[Kbit]のスタティックRAM(SRAM)4つで構成した場合には、スレーブ側のCPU22のメモリマップを例えば図2(A)に示すように構成する。この図2(A)に示すように、スレーブ側のCPU22のメモリ領域(00000(h)~FFFFF(h)番地)に対して、リセット直後にCPU22が読み出すアドレスはFFFF0(h)番地であるため、デュアルポートRAMのアドレスをアドレスデコーダ28によつてFFFF0(h)番地を含む領域(すなわちF0000(h)~FFFFF(h)番地の領域)に割り当てる。

【0020】因みに、図2(A)においては、F0000(h)~FFFFF(h)番地の領域(すなわち64[KByte])をデュアルポートRAM領域としているが、使用するデュアルポートRAMが16[Kbit]のもの2つであるため(すなわち4[KByte])であるため、実際には、図2(B)に示すように、16個(F?000(h)~F?FFF(h)番地; ?=0~F)の4[KByte]の領域に対して2つのデュアルポートRAMを繰り返し割り当てる。またマスタ側のCPU21から見たデュアルポートRAMのアドレスをC0000(h)~C0FFF(h)番地にすれば、マスタ側のアドレスC0XXX(h)番地とスレーブ側のアドレスF?XXX(h)番地(?=0~F)が対応するアドレスになる。従つてスレーブ側のCPU22のブートコードはマスタ側からC0FF0(h)番地に書き込まれる。

【0021】ここでこのような構成を有する複数CPUシステム20においては、図3に示すような処理手順を

7

実行することにより、マスタ側のCPU21からスレーブ側のCPU22を立ち上げると共に、マスタ側からスレーブ側にプログラムコードをダウンロード（すなわち転送）する。まずマスタ側のCPU21は、電源投入後ステップSP1から入って続くステップSP2において、I/Oポート26を制御してスレーブ側のCPU22に対してリセット信号RSTを出力することにより、当該スレーブ側のCPU22をリセット状態にさせる。次にマスタ側のCPU21は、ステップSP3において、ROM24からスレーブ側のCPU22のブートコードを読み出し、デュアルポートRAM23に書き込む。

【0022】この場合、CPU22のブートコードはリセット解除直後に当該CPU22が読みに行く領域に書き込まれる（すなわち上述のメモリマップ例によれば、マスタ側から見てC0FF0(h)番地の領域）。またこのとき書き込まれるブートコードには、スレーブ側のCPU22がデュアルポートRAM23に書き込まれているコードをRAM27にコピーすると共に、コピー終了後にCPU22がそのコピーしたコードに基づいて動作するような情報が含まれている。

【0023】次にマスタ側のCPU21は、ステップSP4においてI/Oポート26を制御してリセット信号RSTを解除し、スレーブ側のCPU22のリセット状態を解除する。そしてマスタ側のCPU21はステップSP5に移り、スレーブ側からの追加コード転送要求を待つ。この状態において、追加コード転送要求が発生すると、マスタ側のCPU21はステップSP6に移り、ROM24から追加コードを読み出して当該追加コードをデュアルポートRAM23に書き込むと共に、スレーブ側のCPU22に対して読み出し要求を出力する。この読み出し要求の出力はデュアルポートRAM23が持つ割り込み発生機能によつて実現される。因みに、この実施例の場合には、デュアルポートRAM23上にフラグを用意しておき、そのフラグを判別することによつて割り込みが読み出し要求であるかそれ以外のものであるかを区別するようになされている。

【0024】次にマスタ側のCPU21は、ステップSP7においてスレーブ側に転送すべきコードの有無を判断し、転送すべきコードがある場合には再びステップSP5に戻つて同様の処理を繰り返し、転送すべきコードがない場合（すなわち転送すべきコードを全て転送し終えた場合）にはステップSP8に移つて転送手順を終了する。因みに、転送手順を終了する場合、最後に転送するコードの後ろに所定のデータを付加することによつて転送手順の終了をスレーブ側のCPU22に対して通知する。

【0025】このようなマスタ側のCPU21の処理に対して、スレーブ側のCPU22は電源投入後ステップSP10から入り、リセット信号RSTの受信によつて

8

続くステップSP11でリセット状態になり、動作を停止する。そしてスレーブ側のCPU22はリセット信号RSTが解除されると続くステップSP12でリセット状態を解除する。そしてスレーブ側のCPU22はステップSP13においてデュアルポートRAM23からブートコードを読み出し（すなわち上述のメモリマップ例によれば、スレーブ側から見てFFFF0(h)番地の領域を読みに行く）、そのブートコードをプログラムとして動作を開始して立ち上がる。この場合、ブートコードには上述したようにスレーブ側のCPU22がデュアルポートRAM23に書き込まれているコードをRAM27にコピーすると共に、コピー終了後CPU22がそのコピーしたコードに基づいて動作するような情報が含まれている。

【0026】このためスレーブ側のCPU22は、続くステップSP14においてデュアルポートRAM23上のコードをRAM27にコピーし、そのRAM27上のコードに基づいて動作する（すなわち動作形態をRAM27上に移す）。次にスレーブ側のCPU22はステップSP15で追加コード転送要求を転送要求待ち状態にあるマスタ側のCPU21に対して出力した後、マスタ側からの読み出し要求を待つ。この場合、転送要求の出力はデュアルポートRAM23が持つ割り込み発生機能によつて実現される。因みに、この実施例の場合には、デュアルポートRAM23上にフラグを用意しておき、そのフラグを判別することによつて割り込みが転送要求であるかそれ以外のものであるかを区別するようになされている。

【0027】スレーブ側のCPU22はマスタ側から読み出し要求が発生すると、続くステップSP16に移り、デュアルポートRAM23上のコードをRAM27にコピーし、そのRAM27上のコードに基づいて動作する。次にスレーブ側のCPU22はステップSP17において受信するコードの有無を判断し、受信するコードがまだある場合には再びステップSP15に戻つて同様の処理を繰り返し、受信するコードがない場合にはステップSP18に移つて処理を終了する。因みに、受信するコードの有無を判断する場合、CPU22は転送されて来たコードに転送終了を意味する所定のコードが或るか否かを見て判断する。

【0028】このような処理手順をマスタ側のCPU21とスレーブ側のCPU22とがそれぞれ実行することにより、ROM24に格納されているスレーブ側のCPU22を動作させるプログラムコードを転送することができる。因みに、マスタ側のCPU21、スレーブ側のCPU22がそれぞれ動作しているときにマスタ側とスレーブ側との間でデータを転送する場合には、上述のようにリセット信号RSTを用いず、デュアルポートRAM23の割り込み発生機能のみによつて行われる。すなわち送信側がデュアルポートRAM23にデータを書き込

んだときに受信側に対して割り込み要求を出力し、受信側はこの割り込み要求に応じてデュアルポートRAM 23に書き込まれているデータを読み出す。これにより各CPU 21、22の動作を止めることなく、各CPU 21、22間でデータを転送することができる。

【0029】以上の構成において、電源投入後、まずマスタ側のCPU 21はI/Oポート26によつてスレーブ側のCPU 22に対してリセット信号RSTを出力し、当該CPU 22をリセット状態にさせる。そしてマスタ側のCPU 21は、CPU 22のリセット状態を維持したままROM 24からCPU 22のブートコードを読み出してデュアルポートRAM 23に書き込む。この場合、ブートコードはリセット解除後にCPU 22が読みに行くデュアルポートRAM 23の領域に対して書き込まれる。ブートコードの書き込みが終了すると、マスタ側のCPU 21はリセット信号RSTを解除してスレーブ側のCPU 22のリセット状態を解除すると共に、追加コードの転送要求待ち状態に入る。

【0030】一方、スレーブ側のCPU 22はリセット信号RSTの解除によつて動作を開始し、デュアルポートRAM 23の所定の領域（すなわちアドレスデコーダ28によつて設定されたブートコードが書き込まれている領域）を読み出してそれを実行する。これによりスレーブ側のCPU 22が立ち上がる。この場合、ブートコードにはデュアルポートRAM 23上のコードをRAM 27にコピーし、コピー終了後そのコピーしたコード上に自身の動作を移す手順が記されているため、CPU 22はこの指示に従つてデュアルポートRAM 23上のコードをRAM 27にコピーすると共に、コピー終了後RAM 27にコピーしたコード上に自身の動作を移す。次にCPU 22は追加コード転送要求をデュアルポートRAM 23を介して転送要求待ち状態にあるマスタ側のCPU 21に対して送出すると共に、読み出し要求待ち状態に入る。

【0031】マスタ側のCPU 21は、この追加コード転送要求に応じてROM 24から追加すべきコードを読み出してデュアルポートRAM 23に書き込むと共に、読み出し要求をデュアルポートRAM 23を介してスレーブ側のCPU 22に対して送出する。スレーブ側のCPU 22は、この読み出し要求に応じてデュアルポートRAM 23上のコードをコピーし、そのコピーしたコード上に自身の動作を移す。そしてCPU 22は、追加するコードがまだある場合には、再び追加コード転送要求を送出し、追加するコードがなくなるまでマスタ側のCPU 21との間で転送手順を繰り返す。そしてスレーブ側のCPU 22は追加するコードがなくなったら転送手順を終了し、RAM 27上のプログラムコードを実行する。

【0032】このようにして複数CPUシステム20では、デュアルポートRAM 23を介してスレーブ側のC

PU 22を立ち上げると共に、デュアルポートRAM 23よりも容量が大きいCPU 22のプログラムコードをスレーブ側に転送することができる。これにより従来必要であつた比較的高価なROMをスレーブ側から削除することができ、全体として構成を簡易にできると共に、コストダウンすることができる。また複数CPUシステム20では、スレーブ側のCPU 22のプログラムコードをマスタ側のROM 24に書き込んでおき、それをスレーブ側に転送するため、特にスレーブ側のCPU 22が増えた場合には、システムのソフトウェア管理を一元化できると共に、ソフトウェアのバージョンアップ時に煩雑なROM交換作業を減らすことができる。

【0033】以上の構成によれば、マスタ側のCPU 21からスレーブ側のCPU 22をリセット状態にするI/Oポート26と、スレーブ側のCPU 22から見たデュアルポートRAM 23のアドレスをブートコード格納アドレスを含む領域に設定するアドレスデコーダ28とを設け、マスタ側のCPU 21からスレーブ側のCPU 22をリセット状態にしてデュアルポートRAM 23にCPU 22のブートコードを書き込んだ後、CPU 22のリセット状態を解除することにより、スレーブ側のCPU 22にブートコードを読み出させて当該スレーブ側のCPU 22を立ち上げることができる。これにより従来必要だつたROMを削除することができ、全体として構成を簡易にできる。

【0034】またブートコード中に、マスタ側からデュアルポートRAM 23に書き込んだコードをRAM 27にコピーする命令を含ませておくことにより、スレーブ側のCPU 22を立ち上げた後、CPU 22を動作させるプログラムコードをマスタ側からスレーブ側に転送することができる。

【0035】なお上述の実施例においては、プロセッサとしてCPU 21、22が用いられたシステムについて述べたが、本発明はこれに限らず、プロセッサとしてDSP (Digital Signal Processor) が用いられたシステムでも良く、要はプロセッサを複数用いたシステムであれば本発明を適用し得る。

【0036】また上述の実施例においては、アドレスデコーダ28をPLDによつて構成した場合について述べたが、本発明はこれに限らず、汎用ロジックを用いて構成しても良い。

【0037】さらに上述の実施例においては、デュアルポートRAM 23を用いてROM 24上のコードをスレーブ側のRAM 27に転送した場合について述べたが、本発明はこれに限らず、転送に際してDMA (Direct Memory Access) を用いても良い。

【0038】また上述の実施例においては、デュアルポートRAM 23のアドレスをF0000(h)~FFFFF(h)番地の領域に割り当てた場合について述べたが、本発明はこれに限らず、リセット直後にCPU 22が読み出すアドレ

11

スを含む領域に設定しさえすれば、デュアルポートRAM 23のアドレスとしては他の領域でも良い。

【0039】さらに上述の実施例においては、マスタ側からスレーブ側にコードを転送する際(図3参照)、転送手順の終了をマスタ側からスレーブ側に通知する場合について述べたが、本発明はこれに限らず、スレーブ側からマスタ側に通知するようにしても良い。

【0040】また上述の実施例においては、不揮発性メモリとしてROM 24を用いた場合について述べたが、本発明はこれに限らず、不揮発性メモリとしてフラッシュメモリ等を用いても良い。

【0041】さらに上述の実施例においては、I/Oポート 26によつてリセット信号RSTを生成する場合について述べたが、本発明はこれに限らず、CPU 21が出力ポートを有するものであれば当該CPU 21でリセット信号RSTを生成するようにしても良い。

【0042】

【発明の効果】上述のように本発明によれば、リセット手段によつてスレーブプロセッサをリセット状態にし、マスタプロセッサによつてメモリからスレーブプロセッサのブートコードを読み出して共有メモリのブートコード格納アドレスに対応する領域に書き込み、スレーブプロセッサのリセット状態を解除することによつてスレーブプロセッサに共有メモリ上のブートコードを読み出させて当該スレーブプロセッサを立ち上げるようにしたことにより、従来スレーブプロセッサのブートコードを格納しておくためスレーブ側に必要だった読み出し専用メモリを削減することができる。かくするにつき一段と構成を簡易にし得る複数プロセッサシステムを実現できる。

【0043】またリセット手段によつてスレーブプロセッサをリセット状態にし、マスタプロセッサによつて第

12

2のメモリからスレーブプロセッサのブートコードを読み出して共有メモリのブートコード格納アドレスに対応する領域に書き込み、スレーブプロセッサのリセット状態を解除することによつてスレーブプロセッサに共有メモリ上のブートコードを読み出させて当該スレーブプロセッサを立ち上げ、さらにスレーブプロセッサにブートコードの命令内容を実行させることによつて、マスタプロセッサが第2のメモリから読み出して共用メモリ上に書き込んだプログラムコードを当該共用メモリから読み出して第1のメモリに格納するようにしたことにより、第2のメモリに格納されたスレーブプロセッサのプログラムコードを第1のメモリにコピーできると共に、従来スレーブプロセッサのブートコードやプログラムコードを格納しておくためスレーブ側に必要だった読み出し専用メモリを削減することができる。かくするにつき一段と構成を簡易にし得る複数プロセッサシステムを実現できる。

【図面の簡単な説明】

【図1】本発明の一実施例による複数CPUシステムの構成を示すブロック図である。

【図2】スレーブ側のCPUのメモリマップを示す略線図である。

【図3】スレーブ側にプログラムコードを転送する際の手順を示すフローチャートである。

【図4】従来の複数CPUシステムの構成を示すブロック図である。

【符号の説明】

1、20……複数CPUシステム、2、21……マスタ側のCPU、3、4、22……スレーブ側のCPU、5、6、23……デュアルポートRAM、7～9、24……ROM、10～12、25、27……RAM、26……I/Oポート、28……アドレスデコーダ。

【図1】

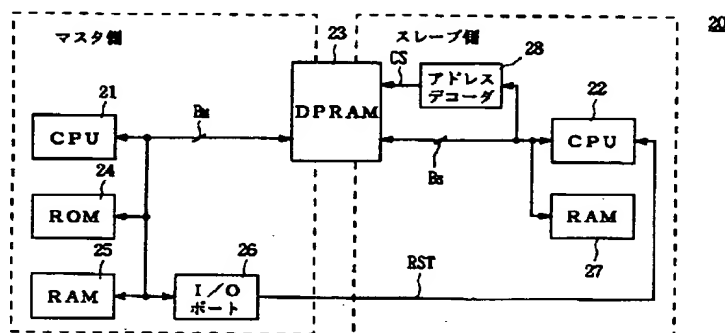


図1 複数CPUシステムの構成

【図2】

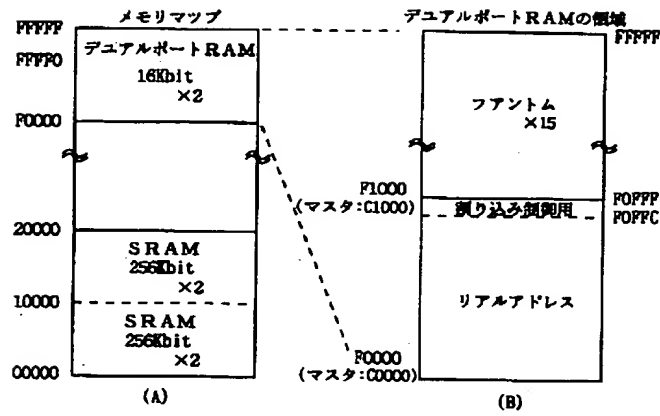


図2 スレーブ側のCPUのメモリマップ

【図4】

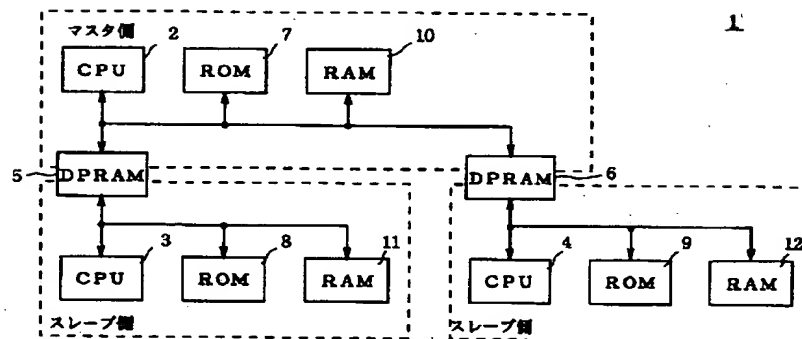


図4 従来の複数CPUシステムの構成

【図3】

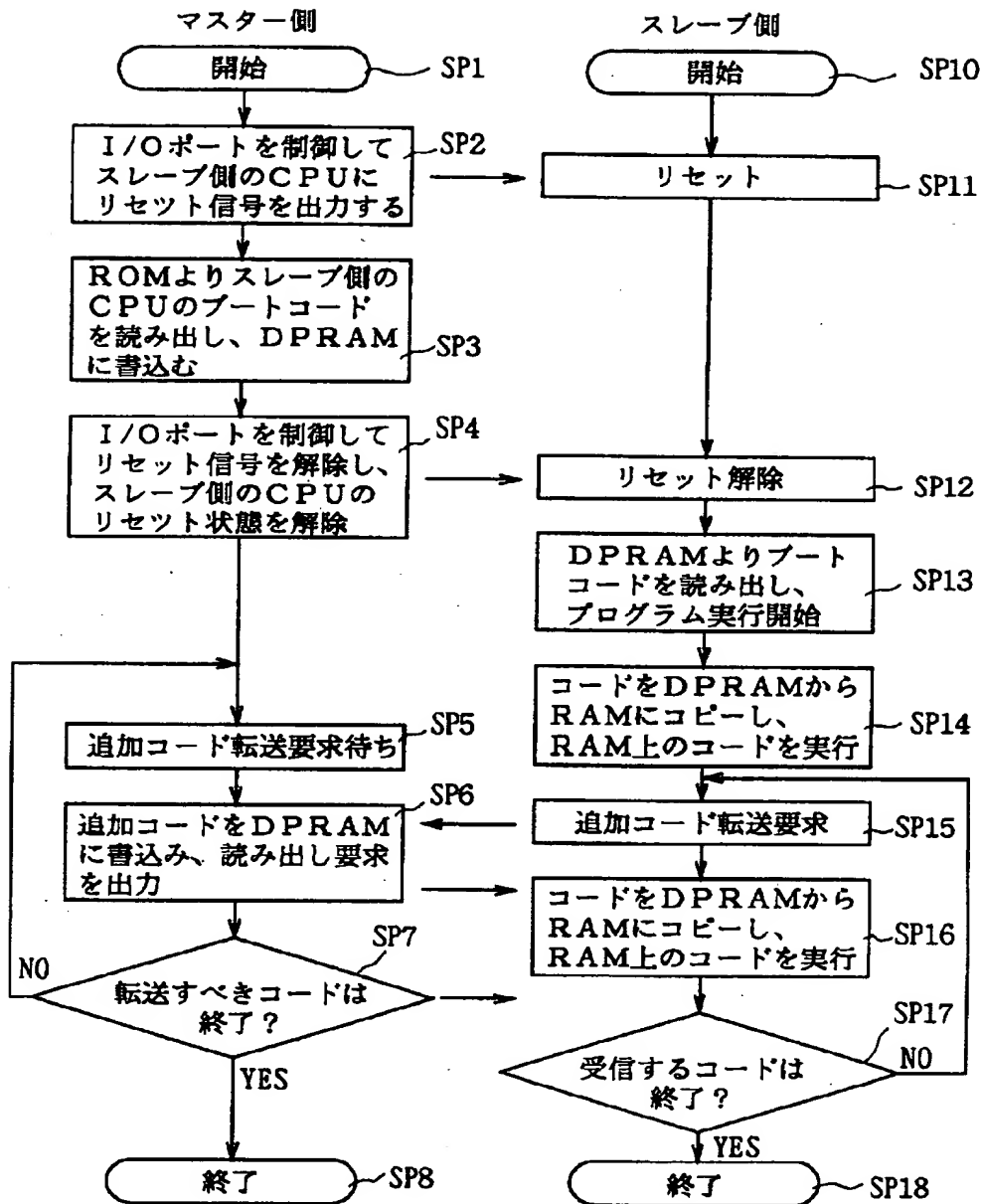


図3 スレーブ側にプログラムコードを転送する手順